Apple Assembly Line

$1.80

Volume 8 -- Issue 3 December, 1987

About Back Issues...

Don't be bashful, just tell me what you need! I have copies of all back issues in stock, at $1.80 each, $18 per volume, or $120 for all seven of the completed volumes. Remember, our volume-year runs from October through September, and I started in 1980.

I also have Quarterly Disks for the entire period, $15 each or four for $60. Each Quarterly Disk covers a calendar quarter, so just specify which months you need. Starting with January 1986 I went to Monthly Disks, so you can get any individual month for only $5 from then till now. Each Quarterly or Monthly Disk contains all of the source code printed in the AAL issues it covers, in the format for the S-C Macro Assembler. Early ones are all DOS, later are split with both DOS and ProDOS directories on the same disk. And since sometime last year we have also been including all of the text files for the articles themselves, as a service to readers who like to have the Echo Speech Synthesizer (or other brands) read it all to them.

Toward a New Standard Assembly Language...

Randy Hyde, who you may remember as author of the Lisa 6502 Assembler, is attempting to organize interested parties to produce a new definitive 65816 assembly language standard. He claims the existing standard, based on Orca, is confusing, overly complex, and idiosyncratic; a new standard could allow assemblers with more power than the Microsoft 8086 assembler to be written for the 65816. Right now Randy is collecting ideas and contacting key individuals (such as the authors of the various existing 65816 assemblers and 65816 books, and the chip's designer), and planning for a conference at WDC in Arizona some time this summer. If you are interested in participating in any way, write to him: Randall Hyde, 65C816 Standards, 2271 Indian Horse Drive, Norco, CA 91760.

Peeking Inside AppleWorks 1.3
+ Subroutine Call Parameter Passage
+ String Handling Subroutines.....Bob Sander-Cederlof

There are a lot of useful subroutines inside AppleWorks. I have been looking at a raw disassembly, and have learned a few new tricks. Even though AppleWorks is ProDOS-based, the subroutines are general enough that you can use them in your own code in any operating system.

All my observations are based on version 1.3, as that is the only one I have. Meanwhile, Apple has moved on to version 2.0 and turned it all over to Claris. That is all right, because I am not proposing that we use the subroutines by loading AppleWorks and calling them; I am proposing that we copy the code or some modification of it into our own programs.

When you boot AppleWorks 1.3 the first thing it does is to copy the APLWORKS.SYSTEM image down from $2000 to $1000. I simply loaded it there from inside the S-C assembler with "BLOAD APLWORKS.SYSTEM,TSYS,A$1000". Then I printed out a huge listing with the monitor's "L" command, and went to work with a pencil. I don't even know what section of AppleWorks I am looking at yet, but it is chock full of interesting code I can use.

The first thing I noticed was that a lot of code did not disassemble correctly: The "L" command went weird after a lot of JSR's. It seems the author liked to call subroutines with parameters in data form following the JSR, like ProDOS MLI calls. In most cases (all I could find) there are either two, four, or six bytes of parameters after these JSR's. The subroutines all call, in turn, on a magic little subroutine which copies the parameter bytes to a standard area in page zero, starting at $9A. This GET.x.PARMS subroutine also updates the return address on the stack so that, when the parameterized call is completed, execution will resume after the parameter bytes.

The GET.x.PARMS subroutine is shown as I found it inside AppleWorks 1.3, in lines 1300-1580. I used the .PH $18AD line to make it look exactly the same as the AppleWorks image. The code looked a little fluffy to me, so I wrote my own version (which is shorter and swifter); you can see it in lines 1830-2070.

The AppleWorks version is evidently one of the busiest pieces of code in the system. I say that, because the author chose to poll the keyboard inside GET.x.PARMS. Line 1550 calls the POLL.KEYBOARD subroutine, which I show with comments in lines 1600-1810. I left this out of my rendition of GET.x.PARMS, because I am building a little package of routines for my own use. I included the listing here because I thought you might like to see how it is done. Notice the buffer holds only ten characters, as written.

Notice that there are three entry points to the GET.x.PARMS subroutine. The first copies four bytes following the JSR to

S-C Macro Assembler Version 2.0			DOS $100, ProDOS $100, both for $120
Version 2.0 DOS Upgrade Kit for 1.0/1.1/1.2 owners			$20
ProDOS Upgrade Kit for Version 2.0 DOS owners			$30
Source Code of S-C Macro 2.0 (DOS or ProDOS)			each, additional $100
S-C DisAssembler (ProDOS only)			without source code $30, with source $50
RAK-Ware DISASM (DOS only)			without source code $30, with source $50
ProVIEW (ProDOS only)			$20
Full Screen Editor for S-C Macro (with complete source code)			$49
S-C Cross Reference Utility			without source code $20, with source $50
S-C Word Processor (with complete source code)			$50
DP18 and DPFP, including complete source and object code			$50
ES-CAPE (Extended S-C Applesoft Program Editor), including Version 2.0 With Source Code			$50
ES-CAPE Version 2.0 and Source Code Update (for Registered Owners)			$30
Bag of Tricks 2 (Quality Software)		($49.95)	$45 *
S-C Documentor (complete commented source code of Applesoft ROMs)			$50
Cross Assemblers for owners of S-C Macro Assembler			$32.50 to $50 each
(Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048, 8051, 8085, 1802/4/5, PDP-11, GI1650/70, Mitsubishi 50740, others)			
Beagle Bros. Applesoft Compiler (ProDOS only)		($74.95)	$65 *
Copy II Plus (Central Point Software)		($39.95)	$30 *
AAL Quarterly Disks		each $15,	or any four for $45

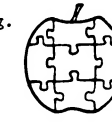
(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

Vinyl disk pages, 6"x8.5", hold two disks each		10 for $6 *
Diskette Mailing Protectors (hold 1 or 2 disks)		40 cents each
(Cardboard folders designed to fit 6"X9" Envelopes.)		or $25 per 100 *
Envelopes for Diskette Mailers		6 cents each
Sider 20 Meg Hard Disk, includes controller & software	($695)	$550 +
Sider 40 Meg Hard Disk, includes controller & software	($995)	$860 +
Minuteman 250 Uninterruptible Power Supply	($359)	$320 +
Minuteman 300 Uninterruptible Power Supply	($549)	$490 +
65802 Microprocessor, 4 MHz (Western Design Center)		$25 *
quikLoader EPROM System (SCRG)	($179)	$170 *
PROmGRAMER (SCRG)	($149.50)	$140 *

"Exploring the Apple IIgs"	Gary B. Little	($22.95)	$21 *
"Apple IIgs Technical Reference"	Michael Fischer	($19.95)	$19 *
"65816/65802 Assembly Language Programming"	Michael Fischer	($21.95)	$20 *
"Programming the 65816"	David Eyes & Ron Lichty	($22.95)	$21 *
"Apple //e Reference Manual"	Apple Computer	($24.95)	$23 *
"Apple //c Reference Manual"	Apple Computer	($24.95)	$23 *
"ProDOS-8 Technical Reference Manual"	Apple Computer	($29.95)	$27 *
"ProDOS-16 Technical Reference Manual"	Apple Computer	($29.95)	$27 *
"Apple IIgs Firmware Reference"	Apple Computer	($24.95)	$23 *
"Apple IIgs Hardware Reference"	Apple Computer	($24.95)	$23 *
"ProDOS Inside and Out"	Dennis Doms & Tom Weishaar	($16.95)	$16 *
"DOStalk Scrapbook"	Tom Weishaar & Bert Kersey	($14.95)	$14 *
"Beneath Apple ProDOS"	Don Worth & Pieter Lechner	($19.95)	$18 *
"Beneath Apple DOS"	Don Worth & Pieter Lechner	($19.95)	$18 *
"Inside the Apple //c"	Gary B. Little	($19.95)	$18 *
"Inside the Apple //e"	Gary B. Little	($19.95)	$18 *
"Understanding the Apple //e"	Jim Sather	($24.95)	$23 *
"Understanding the Apple II"	Jim Sather	($22.95)	$21 *
"Apple II+/IIe Troubleshooting & Repair Guide"	Brenner	($19.95)	$18 *
"Assembly Language for Applesoft Programmers"	Finley & Myers	($18.95)	$18 *
"Now That You Know Apple Assembly Language"	Jules Gilder	($19.95)	$18 *
"Enhancing Your Apple II, vol. 1"	Don Lancaster	($15.95)	$15 *
"Enhancing Your Apple II, vol. 2"	Don Lancaster	($17.95)	$17 *
"Assembly Cookbook for the Apple II/IIe"	Don Lancaster	($21.95)	$20 *
"Microcomputer Graphics"	Roy E.Myers	($14.95)	$14 *
"Assembly Lines -- the Book"	Roger Wagner	($19.95)	$12 *

* These items add $2 for first item, $.75 for each additional item for US shipping.
\+ Inquire for shipping cost.
Customers outside USA inquire for postage needed.
Texas residents please add 8% sales tax to all orders.
<< Master Card, VISA, Discover and American Express >>

S-C Software Corporation
2331 Gus Thomasson #125
DALLAS, TX 75228
Phone 214-324-2050



$9A...9D; the second copies only two bytes; and the third copies any number, which you specify in the A-register. I used a memory search to uncover all the calls on this third entry, and I only found calls which wanted to copy six bytes. There may be others, hidden in other sections of the AppleWorks code.

My more efficient rendition saves one byte by using the BIT opcode ($2C) to skip over the two-byte LDA #2 instruction (see lines 1370 and 1870). I also save by pushing the byte count on the stack instead of saving it in RAM: the storage location is saved, and the PLA is two bytes shorter than a STA. However, I have to pull the byte back off the stack at line 2050, so the net saving is only two bytes. Line 2060, the LDY #0, can be deleted and save another two bytes (Y is already 0, in order for the loop in lines 2010-2040 to terminate). I don't need it because I have not called POLL.KEYBOARD. By the way, we do want to be sure that Y=0 here, because a lot of the subroutines depend on it. If we don't make it one of the functions of GET.x.PARMS, we will have to add a line to most of the subroutines which call GET.x.PARMS.

Inside AppleWorks a lot of string processing goes on. All of the strings I have observed are stored in memory as a length byte followed by the string data bytes. The maximum string will have 255 data bytes. A byte count equal to $00 represents a null string. Lines 2980-3130 are my own code, simply to illustrate how you might code a subroutine to display a string stored in this fashion. Lines 3640-3690 show some strings built by the assembler. In my DEMO code, starting at line 3470, I called on DISPLAY.STRING to print these strings.

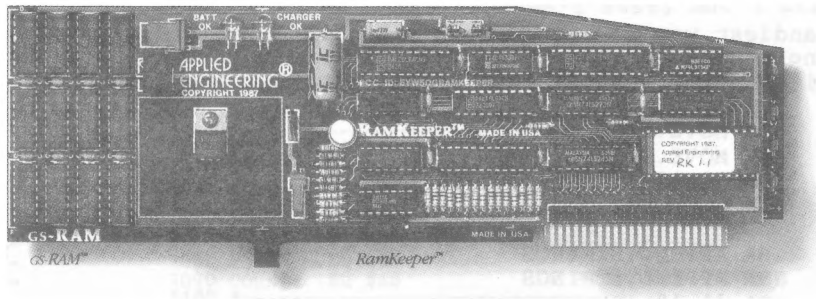
Lines 2080-2970 show my renditions of four subroutines I found inside AppleWorks, which I have named MOVE.STRING, COMPARE.STRING, APPEND.STRINGS, and FILTER.LC.TO.UC. These are not identical to the AppleWorks code, as I found some easy improvements here and there. I showed in comment lines where you can find these subroutines inside the AppleWorks 1.3 image. My DEMO program, lines 3470 to the end, shows some of these in action.

Not all of the subroutines which I found have to do with strings. Lines 3140-3410 show MOVE.BLOCK, which does the equivalent of a monitor "M" command. The three parameters are the destination starting address, the source starting address, and the number of bytes to be moved. (I say "moved", perhaps I should really say "copied".) A similar subroutine which starts immediately after this one, at $1BAC, moves a block of memory "up", by starting at the last byte of the source block and moving backwards. This gives the ability to move a block of memory up to an overlapping area, without clobbering the data.

At the beginning of the code at $1000 there is a JMP table (a long series of JMP xxxx instructions, one right after the other) which is evidently used when other segments want to use some of these general subroutines. Each of the other segments also begins with a JMP table. This is a good scheme for joining together pieces of a large system, and is easy to do. I find it a lot handier than the use of a Link Editor approach,

RamKeeper™

For the "Instant On" Apple IIGS.



Permanent Storage with an "Electronic Hard Disk"

Now when you turn on your IIGS your favorite program can appear on screen in just a few seconds! With RamKeeper, your IIGS memory card will retain stored programs and stored data while your IIGS is turned off. RamKeeper allows you to divide your IIGS memory into part "electronic hard disk" and part RAM for your programs workspace—in almost any way you want and at anytime you want. GS-RAM, GS-RAM Plus, Apple IIGS memory card and most other IIGS memory cards are compatible with RamKeeper.

Supports Up to Two IIGS Memory Cards at the Same Time

If you bought your IIGS with Apple's memory card and later wished you had the GS-RAM, no problem. RamKeeper will support both cards plugged into RamKeeper simultaneously!

How it Works

Just unplug your IIGS memory card from your computer, plug your IIGS memory card into RamKeeper, plug RamKeeper into the IIGS memory slot. If you have another IIGS memory card, an additional card socket on RamKeeper will accommodate your second card. That's all there is to it!

"I've purchased several Applied Engineering products over the years. They're always well made and performed as advertised. I recommend them wholeheartedly."

Steve Wozniak, the creator of Apple Computer

Reliability from the Most Experienced

Applied Engineering has the most experience in the industry with battery-backed memory for the Apple so you are assured of the most reliable memory back-up system available. And in the world of battery-backed memory, reliability is everything! That's why Applied Engineering uses the more dependable Gel-Cell's instead of Ni-Cad batteries (if Ni-Cad's aren't discharged periodically, they lose much of their capacity). RamKeeper has close to 6 times (about 6 hours) the "total power failure" back-up time of other systems. When power returns, RamKeeper automatically re-charges the battery to a full charge. With power from your wall outlet, RamKeeper will back-up your IIGS memory cards RAM indefinately.

RamKeeper Has and Does It All!

- Allows instant access to your programs without slow disk delays
- Configure Kilobytes or Megabytes of instant ROM storage for your favorite programs
- Reduces power strain to your internal IIGS' power supply
- Contains back-up status L.E.D.'s
- Can support up to two IIGS memory cards simultaneously
- Supports both 256K installed memory chip boards like GS-RAM and the Apple IIGS Memory Expansion Card as well as 1 MEG installed memory chip boards like GS-RAM Plus
- 5-year hassle-free warranty
- 15 day money back guarantee
- Proudly made in the U.S.A.
- RamKeeper comes *complete* with battery, software and documentation

Only $179.00!

(GS-RAM card shown in photo not included)

Order Your RamKeeper Today!

See your dealer or call (214) 241-6060, 9:00-11:00 CST, 7 days a week, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 7% sales tax. Add $10.00 if outside U.S.A.

APPLIED ENGINEERING™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

Prices subject to change without notice.

as is used under ProDOS-16. On the other hand, subroutines such as these I have shown in this article are the very type you might want to keep in assembled form as relocatable, linkable, object files, on a library ready to be used by all your future code. Even better, they are the type of subroutines I wish were in the //gs tool boxes, in ROM. And there, they would best be called via a JMP table, for efficiency. These routines are too short to afford the tremendous overhead of "real" toolbox calls.

The handiest way to use subroutines like these would involve writing macros for the calls. For example, here are some macro definitions for MOVE.STRING, APPEND.STRINGS, and MOVE.BLOCK:

```
        .MA MOVE.STRING
        JSR MOVE.STRING
        .DA ]1,]2
        .EM

        .MA APPEND.STRINGS
        JSR APPEND.STRINGS
        .DA ]1,]2
        .EM

        .MA MOVE.BLOCK
        JSR MOVE.BLOCK
        .DA ]1,]2,]3
        .EM
```

These are simple enough, but they can make coding a program easier. If you are error prone, or simply enjoy being cautious, you might add code to the definitions to check for the correct number of macro parameters. For example, MOVE.BLOCK requires three parameters:

```
        .MA MOVE.BLOCK
        .DO ]#=3
        JSR MOVE.BLOCK
        .DA ]1,]2,]3
        .ELSE
==> ERROR:  WRONG NUMBER OF PARAMETERS.  You have ]#, I need 3.
        .FIN
        .EM
```

The line starting "==>" will not be assembled as long as you have 3 parameters. If you have some other number, that line will cause an assembly error, since it starts with an illegal character. This will make it display during assembly, so you can catch and correct your call.

For example, if I try to assemble the following line:

```
1140        >MOVE.BLOCK BUFFER,SIZE
```

you will get the message:

```
>1140 ==> ERROR:  WRONG NUMBER OF PARAMETERS.  You have 2,
I need 3.
```

Did you know you could do that? I wasn't sure, but I tried it and it works.

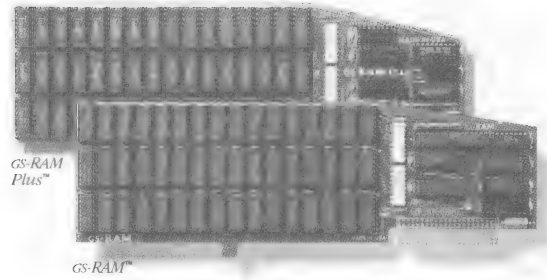
```
                1000 *SAVE S.AW.SUBS
                1010 *-------------------------------
98-             1020 PNTR    .EQ $98,99
9A-             1030 P0      .EQ $9A
9B-             1040 P1      .EQ $9B
9C-             1050 P2      .EQ $9C
9D-             1060 P3      .EQ $9D
9E-             1070 P4      .EQ $9E
9F-             1080 P5      .EQ $9F
A0-             1090 COUNT   .EQ $A0
                1100 *-------------------------------
FD8E-           1110 MON.CROUT   .EQ $FD8E
FDED-           1120 MON.COUT    .EQ $FDED
                1130 *-------------------------------
                1140 *   GET PARMS
                1150 *      (A) = # bytes of parameter info
                1160 *      Copy the bytes to $9A, 9B, ... etc.
                1170 *      Update Return Address
                1180 *      Poll Keyboard for Type-Ahead
                1190 *      Set Y=0, clobbers A and X
                1200 *
                1210 *   For Example:
                1220 *               JSR subroutine
                1230 *               .DA parm1,parm2
                1240 *           <return here>
                1250 *
                1260 *   subroutine JSR GET.4.PARMS
                1270 *              ...
                1280 *              RTS
                1290 *
                1300 *-------------------------------
                1310 *   The following code is as it exists in AppleWorks 1.3
                1320        .PH $18AD
                1330 *-------------------------------
18AD-           1340 AW.GET.PARM.TEMP .BS 1
                1350 *-------------------------------
18AE- A9 04     1360 AW.GET.4.PARMS LDA #4
18B0- D0 02     1370                BNE AW.GET.A.PARMS
18B2- A9 02     1380 AW.GET.2.PARMS LDA #2
18B4- A8        1390 AW.GET.A.PARMS TAY
18B5- 8D AD 18  1400        STA AW.GET.PARM.TEMP
18B8- BA        1410        TSX
18B9- BD 03 01  1420        LDA $0103,X
18BC- 85 98     1430        STA PNTR
18BE- 18        1440        CLC
18BF- 6D AD 18  1450        ADC AW.GET.PARM.TEMP
18C2- 9D 03 01  1460        STA $0103,X
18C5- BD 04 01  1470        LDA $0104,X
18C8- 85 99     1480        STA PNTR+1
18CA- 69 00     1490        ADC #0
18CC- 9D 04 01  1500        STA $0104,X
18CF- B1 98     1510 .1     LDA (PNTR),Y
18D1- 99 99 00  1520        STA PNTR+1,Y
18D4- 88        1530        DEY
18D5- D0 F8     1540        BNE .1
18D7- 20 A7 1F  1550        JSR POLL.KEYBOARD
18DA- A0 00     1560        LDY #0
18DC- 60        1570        RTS
                1580        .EP
```

```
                    1590 *--------------------------------
                    1600 *   POLL KEYBOARD
                    1610 *--------------------------------
                    1620         .PH $1FA7
                    1630 POLL.KEYBOARD
1FA7- AD 00 C0      1640         LDA $C000     ANY KEY PRESSED?
1FAA- 10 24         1650         BPL .3        ...NO, RETURN NOW
1FAC- 8D 10 C0      1660         STA $C010     ...YES, CLEAR STROBE
1FAF- AE 61 C0      1670         LDX $C061     OPEN APPLE PRESSED?
1FB2- 30 07         1680         BMI .1        ...YES
1FB4- AE 62 C0      1690         LDX $C062     SOLID APPLE PRESSED?
1FB7- 30 02         1700         BMI .1        ...YES
1FB9- 29 7F         1710         AND #$7F      ...NO APPLES, SO CLEAR BIT 7
1FBB- AE 84 11      1720 .1      LDX $1184 <<KEY.BUFFER.INDEX>>
1FBE- 9D 7A 11      1730         STA $117A,X   <<KEY.BUFFER>>
1FC1- E8            1740         INX
1FC2- E0 0A         1750         CPX #10       AT END OF BUFFER YET?
1FC4- 90 02         1760         BCC .2        ...NO END YET
1FC6- A2 00         1770         LDX #0        ...END, SO WRAP AROUND
1FC8- EC 85 11      1780 .2      CPX $1185 <<KEY.BUFFER.OUTDEX>>
1FCB- F0 03         1790         BEQ .3        BUFFER IS FULL
1FCD- 8E 84 11      1800         STX $1184 <<KEY.BUFFER.INDEX>>
1FD0- 60            1810 .3      RTS
                    1820         .EP
                    1830 *--------------------------------
                    1840 *   My More Efficient Version
                    1850 *--------------------------------
085A- A9 04         1860 GET.4.PARMS LDA #4
085C- 2C            1870         .HS 2C     SKIP NEXT 2 BYTES
085D- A9 02         1880 GET.2.PARMS LDA #2
085F- A8            1890 GET.A.PARMS TAY
0860- 48            1900         PHA
0861- BA            1910         TSX
0862- BD 04 01      1920         LDA $0104,X  GET RETURN ADDR-LO
0865- 85 98         1930         STA PNTR      KEEP FOR VECTOR
0867- 18            1940         CLC
0868- 7D 01 01      1950         ADC $0101,X  ADD # BYTES
086B- 9D 04 01      1960         STA $0104,X  UPDATE RETURN ADDR-LO
086E- BD 05 01      1970         LDA $0105,X  GET RETURN ADDR-HI
0871- 85 99         1980         STA PNTR+1      SAVE FOR VECTOR
0873- 69 00         1990         ADC #0
0875- 9D 05 01      2000         STA $0105,X  UPDATE RETURN ADDR-HI
0878- B1 98         2010 .1      LDA (PNTR),Y  USING VECTOR, COPY PARMS
087A- 99 99 00      2020         STA PNTR+1,Y     ...TO 9A,9B,etc.
087D- 88            2030         DEY
087E- D0 F8         2040         BNE .1
0880- 68            2050         PLA           GET LENGTH OFF STACK
0881- A0 00         2060         LDY #0
0883- 60            2070         RTS
                    2080 *--------------------------------
                    2090 *   MOVE STRING
                    2100 *      JSR MOVE.STRING
                    2110 *      .DA destination,source
                    2120 *   (at $1EF8 in AppleWorks 1.3)
                    2130 *--------------------------------
                    2140 MOVE.STRING
0884- 20 5A 08      2150         JSR GET.4.PARMS
0887- B1 9C         2160         LDA (P2),Y    COPY THE LENGTH BYTE
0889- 91 9A         2170         STA (P0),Y
088B- F0 08         2180         BEQ .2        STRING IS EMPTY
088D- A8            2190         TAY           LENGTH TO Y
088E- B1 9C         2200 .1      LDA (P2),Y
0890- 91 9A         2210         STA (P0),Y
0892- 88            2220         DEY
0893- D0 F9         2230         BNE .1
0895- 60            2240 .2      RTS
                    2250 *--------------------------------
                    2260 *   COMPARE TWO STRINGS
                    2270 *      JSR COMPARE.STRINGS
                    2280 *      .DA str1,str2
                    2290 *    return Carry Clear if str1 < str2; else Carry Set
                    2300 *   (at $1ED9 in AppleWorks 1.3)
                    2310 *--------------------------------
                    2320 COMPARE.STRINGS
0896- 20 5A 08      2330         JSR GET.4.PARMS
0899- B1 9A         2340         LDA (P0),Y    GET LENGTH OF SHORTER STRING
089B- D1 9C         2350         CMP (P2),Y
089D- 90 02         2360         BCC .1
```

```
089F- B1 9C     2370          LDA (P2),Y
08A1- AA        2380 .1       TAX             LENGTH OF SHORTEST TO X-REG
08A2- F0 0A     2390          BEQ .3          SHORTEST IS NULL
08A4- C8        2400 .2       INY             COMPARE BODY OF STRINGS
08A5- B1 9A     2410          LDA (P0),Y
08A7- D1 9C     2420          CMP (P2),Y
08A9- D0 09     2430          BNE .4          NOT SAME, SO CARRY GIVES RELATION
08AB- CA        2440          DEX
08AC- D0 F6     2450          BNE .2          MORE TO COMPARE
08AE- A0 00     2460 .3       LDY #0          STRINGS MATCH TO END OF SHORTEST
08B0- B1 9A     2470          LDA (P0),Y      COMPARE ON BASIS OF LENGTH
08B2- B1 9C     2480          LDA (P2),Y
08B4- 60        2490 .4       RTS
                2500 *--------------------------------
                2510 *    APPEND TWO STRINGS
                2520 *       JSR APPEND.STRINGS
                2530 *       .DA stringA,stringB
                2540 *    (at $1341 in AppleWorks 1.3)
                2550 *--------------------------------
                2560 APPEND.STRINGS
08B5- 20 5A 08  2570          JSR GET.4.PARMS
08B8- B1 9A     2580          LDA (P0),Y      GET LENGTH OF STRING A
08BA- 48        2590          PHA             SAVE IT
08BB- 18        2600          CLC
08BC- 71 9C     2610          ADC (P2),Y      ADD LENGTH OF STRING B
08BE- 91 9A     2620          STA (P0),Y      MAKES LENGTH OF COMBINED STRING
08C0- 68        2630          PLA             GET LENGTH OF STRING A AGAIN
08C1- 18        2640          CLC
08C2- 65 9A     2650          ADC P0          BUMP POINTER TO END OF STRING A
08C4- 85 9A     2660          STA P0
08C6- 90 02     2670          BCC .1
08C8- E6 9B     2680          INC P1
08CA- B1 9C     2690 .1       LDA (P2),Y      LENGTH OF STRING B
08CC- F0 08     2700          BEQ .3          ...NULL, SO FINISHED
08CE- A8        2710          TAY
08CF- B1 9C     2720 .2       LDA (P2),Y
08D1- 91 9A     2730          STA (P0),Y
08D3- 88        2740          DEY
08D4- D0 F9     2750          BNE .2
08D6- 60        2760 .3       RTS
                2770 *--------------------------------
                2780 *    FILTER LOWER CASE to UPPER CASE in a STRING
                2790 *       JSR FILTER.LC.TO.UC
                2800 *       .DA string
                2810 *    (at $1BF in AppleWorks 1.3)
                2820 *--------------------------------
                2830 FILTER.LC.TO.UC
08D7- 20 5D 08  2840          JSR GET.2.PARMS
08DA- B1 9A     2850          LDA (P0),Y      GET LENGTH OF STRING
08DC- F0 12     2860          BEQ .3          NULL STRING
08DE- A8        2870          TAY             LENGTH TO Y-REG
08DF- B1 9A     2880 .1       LDA (P0),Y
08E1- C9 61     2890          CMP #'a'
08E3- 90 08     2900          BCC .2
08E5- C9 7B     2910          CMP #'z'+1
08E7- B0 04     2920          BCS .2
08E9- 29 DF     2930          AND #$DF        TURN OFF LOWER/CASE BIT
08EB- 91 9A     2940          STA (P0),Y
08ED- 88        2950 .2       DEY
08EE- D0 EF     2960          BNE .1          ...MORE BYTES
08F0- 60        2970 .3       RTS
                2980 *--------------------------------
                2990 *    DISPLAY STRING
                3000 *       JSR DISPLAY.STRING
                3010 *       .DA string.address
                3020 *--------------------------------
                3030 DISPLAY.STRING
08F1- 20 5D 08  3040          JSR GET.2.PARMS
08F4- B1 9A     3050          LDA (P0),Y      GET LENGTH
08F6- F0 0C     3060          BEQ .2          ...NULL STRING
08F8- 85 A0     3070          STA COUNT
08FA- C8        3080 .1       INY
08FB- B1 9A     3090          LDA (P0),Y
08FD- 20 ED FD  3100          JSR MON.COUT
0900- C4 A0     3110          CPY COUNT
0902- 90 F6     3120          BCC .1
0904- 60        3130 .2       RTS
```

For Those Who Want the Most. From Those Who Make the Best. GS-RAM.™

Now expand the IIGS' RAM and ROM with up to 8 MEG of "Instant On" memory with the all new GS-RAM!



GS-RAM has an all new design. A design that delivers higher performance including increased speed, greater expandability, and improved software.

More Sophisticated, Yet Easier to Use

GS-RAM works with all IIGS software. In fact *any* program that runs on Apple's smaller memory card runs on the GS-RAM. But with GS-RAM you can have more memory, improved performance, and almost unlimited expansion capabilities. We've designed the new GS-RAM to be easier to use too—you don't have to adjust the size of your RAM disk every time you use a DMA device. No other RAM card with more than 4 banks of memory installed can make the same claim.

More than Just Hardware

Each GS-RAM and GS-RAM Plus includes the most powerful set of IIGS software enhancements available anywhere. In fact, our nearest competitor offers only a fraction of the invaluable programs that we include with each GS-RAM card. This software includes the most powerful disk-caching program available, the GS-RAM Cache. The Cache will make most of your applications run up to 7 *times faster.* Also included is a diagnostic utility that lets you test your GS-RAM by graphically showing the location of any bad or improperly installed RAM chips. And for AppleWorks users, we give you our exclusive Expander program that dramatically enhances both the capabilities and speed of AppleWorks.

Making AppleWorks Even Better

Applied Engineering's Expander program eliminates AppleWorks internal memory limits allowing it to recognize up to 8 megabytes of desktop workspace. You can increase the limits from only 7,250 lines to 22,600 lines in the word processor and from 6,350 records to 22,600 records in the database. The Expander allows all of AppleWorks, including print functions, to automatically load into RAM. The clipboard size will increase from 255 to 2,042 lines maximum. GS-RAM will automatically segment larger files so you can save them onto multiple floppies. And GS-RAM provides a built-in print buffer that allows you to continue working in AppleWorks while your printer is still processing text. You can even load Pinpoint or MacroWorks and your favorite spelling checker into RAM for instant response.

Grow by Kilobytes or Megabytes

We offer GS-RAM in two configurations so you can increase your memory 256K at a time (GS-RAM) or a megabyte at a time (GS-RAM Plus). Both are IIGS compatible and both come with our powerful enhancement software. GS-RAM can hold up to 1.5 MEG of 256K chips and GS-RAM Plus can hold up to 6 MEG using 1 MEG chips. And since both use standard RAM chips (not high-priced SIMM's), you'll find expanding your GS-RAM or GS-RAM Plus easy, convenient, and very economical. For further expansion, you can plug a 2 MEG "piggyback" card into the GS-RAM's expansion port for up to 3.5 MEG of total capacity. Or up to a whopping 8 MEG on GS-RAM Plus. If a GS-RAM owner outgrows 3.5 MEG, he can easily upgrade to GS-RAM Plus for a nominal charge.

Permanent Storage for an "Instant On" Apple

With our RamKeeper™ back-up option, your GS-RAM or GS-RAM Plus will retain both programs and data while your IIGS is turned off! Now when you turn your IIGS back on, your favorite software is on your screen in under 4 seconds! With RamKeeper you can divide your IIGS memory into part "electronic hard disk," and part extended RAM. Even change the memory boundaries at any time—and in any way—you want. Because Applied Engineering has the most experience in the industry with battery-backed memory for the Apple, you are assured of the most reliable memory back-up system available. And in the world of battery-backed memory, *Reliability* is everything. That's why Applied Engineering uses state-of-the-art "GEL-CELL's" instead of Ni-Cad batteries (if Ni-Cads aren't discharged periodically, they lose much of their capacity). RamKeeper has about 6 hours of "total power failure" back-up time. That's 6 times the amount of other systems. But with power from your wall outlet, RamKeeper will back-up GS-RAM, GS-RAM Plus, or most other IIGS memory cards indefinitely. Should you ever have a "total power failure," RamKeeper switches to its 6-hour battery. When power returns, RamKeeper will automatically recharge the battery to full power. RamKeeper incorporates a dual-rate charger, status L.E.D.'s, and advanced power reducing circuitry. RamKeeper comes *complete* with battery, software, and documentation.

"In quality, performance, compatibility, expandability and support, Applied Engineering's GSRAM and GSRAM Plus are number one."

Steve Wozniak, the creator of Apple Computer

GS-RAM's Got it ALL!

- 5-year warranty — parts & labor
- 6 RAM banks (most cards have 4)
- Memory expansion port
- ROM expansion port
- Ultra-fast disk caching on ProDOS 8 *AND* ProDOS 16.
- Expands AppleWorks internal limits
- Includes hi-res self test
- No soldered-in RAM chips
- Expandable to 8 MEG
- No configuration blocks to set
- RamKeeper back-up option allows permanent storage of programs & data
- 15-day money-back guarantee
- Proudly made in the U.S.A.

GS-RAM with 256K	**$189**
GS-RAM with 512K	**$259**
GS-RAM with 1 MEG	**$399**
GS-RAM with 1.5 MEG	**$539**
GS-RAM with 2.5 to 3.5 MEG	**CALL**
GS-RAM Plus with 1-8 MEG	**CALL**
RamKeeper Option	**$179**

Order today!

See your dealer or call Applied Engineering today, 9 a.m. to 11 p.m. 7 days. Or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 7% sales tax. Add $10.00 outside U.S.A.

APPLIED ENGINEERING™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

Prices subject to change without notice.

GS-RAM, GS-RAM Plus and RamKeeper are trademarks of Applied Engineering. Other brands and product names are registered trademarks of their respective holders.

```
                     3140 *--------------------------------
                     3150 *   MOVE MEMORY BLOCK
                     3160 *      JSR MOVE.BLOCK
                     3170 *      .DA destination,source,num.bytes
                     3180 *   (at $1B84 in AppleWorks 1.3)
                     3190 *--------------------------------
                     3200 MOVE.BLOCK
0905- A9 06          3210        LDA #6        GET 6 PARM BYTES
0907- 20 5F 08       3220        JSR GET.A.PARMS
090A- A5 9F          3230        LDA P5        GET NUMBYTES-HI (# FULL PAGES)
090C- F0 0F          3240        BEQ .2        ...NO FULL PAGES TO MOVE
                     3250 *---Move Full Pages-------------
090E- B1 9C          3260 .1     LDA (P2),Y
0910- 91 9A          3270        STA (P0),Y
0912- C8             3280        INY
0913- D0 F9          3290        BNE .1        ...UNTIL FULL PAGE MOVED
0915- E6 9B          3300        INC P1        SOURCE-HI
0917- E6 9D          3310        INC P3        DESTINATION-HI
0919- C6 9F          3320        DEC P5        # FULL PAGES LEFT
091B- D0 F1          3330        BNE .1        ...STILL MORE
                     3340 *---Move Partial Page-----------
091D- C4 9E          3350 .2     CPY P4        FINISHED PARTIAL PAGE?
091F- F0 07          3360        BEQ .3        ...YES
0921- B1 9C          3370        LDA (P2),Y
0923- 91 9A          3380        STA (P0),Y
0925- C8             3390        INY
0926- D0 F5          3400        BNE .2        ...ALWAYS
0928- 60             3410 .3     RTS
                     3420 *--------------------------------
                     3430 *
                     3440 *   DEMONSTRATION OF SOME STRING SUBROUTINES
                     3450 *
                     3460 *--------------------------------
0929- 20 F1 08       3470 DEMO   JSR DISPLAY.STRING
092C- 55 09          3480        .DA STR.A
092E- 20 8E FD       3490        JSR MON.CROUT
0931- 20 F1 08       3500        JSR DISPLAY.STRING
0934- 67 09          3510        .DA STR.B
0936- 20 8E FD       3520        JSR MON.CROUT
0939- 20 84 08       3530        JSR MOVE.STRING
093C- 79 09 67
093F- 09             3540        .DA STR.C,STR.B
0940- 20 F1 08       3550        JSR DISPLAY.STRING
0943- 79 09          3560        .DA STR.C
0945- 20 8E FD       3570        JSR MON.CROUT
0948- 20 B5 08       3580        JSR APPEND.STRINGS
094B- 79 09 55
094E- 09             3590        .DA STR.C,STR.A
094F- 20 F1 08       3600        JSR DISPLAY.STRING
0952- 79 09          3610        .DA STR.C
0954- 60             3620        RTS
                     3630 *--------------------------------
0955- 11             3640 STR.A  .DA #SZ.A
0956- D4 C8 C9
0959- D3 A0 C9
095C- D3 A0 D3
095F- D4 D2 C9
0962- CE C7 A0
0965- C1 AE          3650        .AS -/THIS IS STRING A./
11-                  3660 SZ.A   .EQ *-STR.A-1
0967- 11             3670 STR.B  .DA #SZ.B
0968- D4 C8 C9
096B- D3 A0 C9
096E- D3 A0 D3
0971- D4 D2 C9
0974- CE C7 A0
0977- C2 AE          3680        .AS -/THIS IS STRING B./
11-                  3690 SZ.B   .EQ *-STR.B-1
0979-                3700 STR.C  .BS 80
                     3710 *--------------------------------
```

Screen Dump PLUS!.....Louis Pitz

Enclosed is SCRNDUMP.PLUS, an enhancement to Steve Knouse's Generic Screen Dump in Apple Assembly Line, September 1983. The main enhancements are 40/80/Lores capability and, via conditional assembly, either a DOS 3.3 or a ProDOS version. The Lores capability is modified from a routine by R.M. Mottola in Nibble/#3/p.18.

The idea is to squeeze in as many features as possible and still have a utility that will fit in good old page 3 space ($300.3CF). However, there are times you have something else in page 3 and need your screendump utility elsewhere. This is easy to do just by reassembling the screendump at another address. However, I'm lazy, and when I need a screendump utility installed I don't want to have to hunt for my assembler disk. So this version is self-relocating in that you can BLOAD and CALL to install it at any address where you have 284 bytes ($11C) free.

This is too big to fit in page 3, so I borrowed Bill Morgan's idea from AAL/Nov.82, to use the upper part of page 2. The input trap and dump portion of SCRNDUMP.PLUS fits in page 3, while the installation takes the top of page 2. Since installation is a one-time affair, it's disposable, although it will remain there if you don't make too heavy use of the input buffer.

In addition to Steve Knouse and R.M. Mottola, I also learned and borrowed from Bill Parker's EPSON TEXT SCREEN DUMP, 1982 and Gary Little's ProDOS DUMP# utility in A+/Jul.85/p.69. I give credit to Roger Wagner's "Assembly Lines: The Book" for relocation ideas. There are a lot of neat techniques in this short utility, most of course 'borrowed' from other sources.

Now for some comments on why I chose BLOAD & CALL logic. You can install the dump via BRUN in immediate mode in DOS 3.3. However, readers of AAL know (AAL/JUN.86, AUG.86, SEP.86) that there are problems in DOS 3.3 with BRUN from within an Applesoft program. So I chose BLOAD & CALL to avoid the extra code you'd need to solve the problem. The BLOAD and CALL 41876 approach of AAL/AUG.86 is a good alternative to BRUN.

However, 3.3 DOSologists need not be ashamed, for ProDOS has its quirks also! See Call-Apple/Apr.84/p.39/Cecil Fretwell for details, but in ProDOS you must choose between a BLOAD & CALL or BRUN approach even for immediate mode! Replace Lines 2120 & 2130 by

```
2120  STX KSWL
2130  STY KSWH
2134  RTS
2138  .BS 2  (filler)
```

for BRUN logic. Again, I chose BLOAD & CALL logic for both DOS 3.3 and ProDOS to be consistent. Also, you can CALL 692 (or 2B4G from monitor) to rehook if you haven't made too heavy use of the input buffer.

Now Apple speaks IBM. Three times faster than IBM.

Introducing PC Transporter.™ The Apple® II expansion board that lets you run MS®-DOS programs.

Now your Apple II can run over 10,000 programs you could never use before. Like Lotus® 1-2-3® MultiMate® dBASE III PLUS® Even Flight Simulator®

With PC Transporter, MS-DOS programs run on your Apple II like they do on IBM® PC's or compatibles. With one important difference. PC Transporter runs most of those programs *three times faster* than an IBM PC/XT®

Plus, to speed through number-crunching tasks, you can use our optional 8087-2 math co-processor chip. It plugs into a socket on the PC Transporter.

Less expensive than an IBM clone.

Sure, a stripped-down IBM

clone costs about the same as the PC Transporter. But the peripherals it takes to get the clone up and running make the clone cost about three times what our American-made card costs.

You don't have to buy new hardware to use PC Transporter.

Works with the hardware you already own.

With PC Transporter, MS-DOS programs see your Apple hardware as IBM hardware. You can use the same hardware you have now.

With IBM software, your Apple hardware works just like IBM hardware. Including your drives, monitors, printers, printer cards, clock cards and serial clocks.

You can use your IIe® or IIGS™ keyboard with IBM software. Or use our optional IBM-style keyboard (required for the II Plus).

You can use your Apple mouse. Or an IBM compatible serial mouse.

Plenty of power.

PC Transporter gives you as much as 640K of user RAM and 128K of system RAM in the IBM mode.

PC Transporter also is an Apple expansion card, adding up to 768K of extra RAM in the Apple mode. The Apple expansion alone is a $300 value.

Easy to install.

You can install PC Transporter in about 15 minutes, even if you've never added an expansion board. You don't need special tools. Simply plug it into an Apple expansion slot (1 through 7 except 3), connect a few cables and a disk drive, and go!

PC Transporter taps into the world's largest software library. Now your Apple can run most of the IBM software you use at work. And it opens a new world of communications programs, games and bulletin boards.

A universal disk drive controller.

PC Transporter supports 3.5" and 5.25" MS-DOS and ProDOS formatted diskettes. You'll shift instantly between Apple ProDOS and IBM MS-DOS.

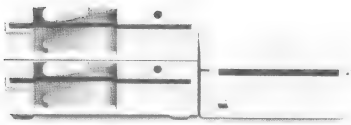
You'll need our versatile 5.25" 360K drive system to run IBM applications from 5.25" floppy disks. Use your Apple 5.25" drive for Apple 5.25" disks.

An Apple Disk 3.5 Drive will support the new 3.5"disks whether they're IBM MS-DOS formatted or Apple ProDOS formatted. The PC Transporter acts like an Apple Disk 3.5 Drive disk controller for IIGS, IIe, and II Plus users.

PC Transporter supports up to 5 drives in a number of combinations.

For example, you can connect a 5.25 Applied Engineering 360K dual-drive system directly to the card. Then plug two daisy-chained Apple 3.5 Drives (not the Apple UniDisk 3.5) to the dual-drive system. For a fifth drive, use a ProDOS file as an IBM hard disk.

PC Transporter controls Apple and IBM compatible disk drives. It supports 3.5" and 5.25" MS-DOS and ProDOS formatted diskettes.



Versatile data storage.

PC Transporter reads MS-DOS and translates it into Apple native ProDOS. You can store IBM programs and data on any ProDOS storage device including the Apple 3.5 Drive, Apple UniDisk™ 3.5, Apple 5.25" drive, SCSI or ProDOS compatible hard drives. (You can use the Apple UniDisk 3.5 with its own controller card for storing programs and data,but not for directly booting an IBM formatted disk.)

You can even use our 360K PC compatible drive for ProDOS

Make your Apple speak IBM.

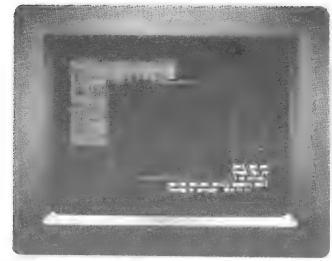
PC Transporter memory choices.

RAM in Apple mode:	RAM in IBM mode:	Price:
384K	**256K**	**$489.00**
512K	**384K**	**529.00**
640K	**512K**	**569.00**
768K	**640K**	**609.00**

Note: The IBM mode is 128K less because the PC Transporter uses 128K for system memory.

IIGS Installation Kit	**49.00**
IIe/II Plus Installation Kit	**39.00**
PC Transporter Accessories	
5.25" IBM Format 360K Drive Systems	
Single-Drive System	**269.00**
Dual-Drive System	**399.00**
Half-Height Drive (add to Single-Drive system to make Dual-Drive)	**135.00**
IBM-Style Keyboard (required for Apple II Plus. Requires IBM Keyboard Cable.)	**139.00**
IBM Keyboard Cable	**34.00**
Sony RGB Monitor	**499.00**
Analog RGB Cable (for use with Sony monitor)	**39.00**
Digital RGB Cable (for use with Sony monitor)	**39.00**
Digital RGB Adapter	**24.00**
ColorSwitch (included with IIGS Installation Kit)	**44.00**
128K ZIP PC Transporter Memory Expansion Chip Set	**40.00 per set**
8087-2 Math Co-processor Chip	**229.00**
Heavy Duty Power Supply (IIe and II Plus only)	**69.00**

See your dealer or call or send check or money order to Applied Engineering. MasterCard, VISA and COD welcome. Texas residents add 6¼% sales tax.



PC Transporter produces better IBM graphics than IBM. Analog is sharper than digital. So with an analog RGB monitor, PC Transporter's CGA graphics and text are superior to IBM's digital display — even while running IBM software!

And, you can also use an Apple composite monitor in IBM text or graphics mode.

storage and a 143K Apple 5.25" drive for MS-DOS storage.

Created by Apple's original designers.

The brains behind PC Transporter were also behind your Apple II.

The PC Transporter design team includes the former project managers for the creation of the Apple IIe and IIc. The co-designer of the Apple II disk controller. And the first full-time Apple programmer and author of the ProDOS operating system.

So you know the PC Transporter and your Apple were made for each other.

Support and service from the leader in Apple add-ons.

Applied Engineering sells more Apple peripheral boards than anyone else — including Apple Computer. So you know we'll be around after the sale.

PC Transporter comes with a 15-day money back guarantee. If you're not fully satisfied after using it, return it for a full refund. PC Transporter also comes with a 1-year warranty.

How to get your PC Transporter today.

See your dealer. Or call Applied Engineering any day between 9 a.m. and 11 p.m. CST at 214-241-6060.

Applied Engineering

The Apple enhancement experts.

P.O. Box 798, Carrollton, TX 75006
214-241-6060
A Division of AE Research Corporation

Now to describe the utility in more detail. Lines 1540-1630 find where it has been BLOADed. I think it's a good idea to inhibit interrupts while getting information from the stack. Then lines 1640-1720 modify (SETRAP+1) and (SETRAP+3) to point at TRAP. Lines 1730-1900 put the current INPUT address at TRAP+1.TRAP+2 and TRAP+8.TRAP+10: the first so we can 'daisy-chain' our input TRAP at the end of the current INPUT routine, and the second so we can terminate via CTRL-T to remove our INPUT trap. Lines 1910-1980 set default parameters at locations 6-9 and $ED. If you BLOAD without CALL-ing, you must set these parameters yourself. As far as I can tell, there are 11 page 0 locations not used by Applesoft, Integer Basic, DOS 3.3, ProDOS (including interrupt handlers), or the System Monitor. They are 6-9, $1E, $EB-EF, and $F9.

Anyway, 6 (PRFLAG) is used as an enable/disable (1/0) flag- for when you want to disable dumping without removing the input trap. 7 (WINTOP) and 8 (WINBOT) are used to set a 'window' to dump if you don't want to dump the whole screen. Here the top line is 0 and bottom is 23 ($17). Note that I couldn't fit left and right margin windows in. If you want a routine that does both 40 and 80 column dumps you'll have to be careful in adding left and right margins. $ED (FLAG) is used to flag if a dump has been done via CTRL-P from the TRAP (value $FF) or just a CALL 802 (JSR $322) (value 0).

SETRAP, Lines 2050-2150, hooks TRAP into DOS or ProDOS.

TRAP, our input 'filter', does its thing by first daisy-chaining to the current input routine and letting it do all the dirty work. Only then does TRAP check for either CTRL-T (Terminate TRAP and reset input hooks) or CTRL-P (Print the screen). If neither, exit to caller of input routine. If a CTRL-P, Set FLAG ($ED) to $FF to show we came thru TRAP. This avoids a JSR DUMP, which would not be relocatable. You could have a JSR DUMP with address filled in during the first portion of the utility just like SETRAP+1, +3 are filled in from Lines 1640-1720. That way you wouldn't need FLAG at all, but the approach takes 3 more bytes. Roger Wagner's book has other ways to do a relocatable JSR, but remember that we want a DUMP that can be called either directly or from TRAP.

Note that lines 2280 and 2290 inside TRAP avoid another ProDOS 'gotcha'. A JMP $3D0 (DOS.WARM) in ProDOS (Q from IIgs Monitor) returns to immediate mode but erases Applesoft variables, unlike the friendly DOS 3.3! Thanks to Steve and Marsha Meuse for this one (Nibble/Nov.85/p.20). So from ProDOS JSR RSTINT and JSR BASIC2 exit without erasing variables. BASIC2 ($E003) is the address a CTRL-C <Return> from monitor uses. RSTINT ($9A17) is the same in BASIC.SYSTEM versions 1.0 and 1.1, the only ones so far. Future versions may change, so beware.

For the DUMP itself, we disable interrupts and save the A-X-Y registers, the cursor location CH (so after the dump is done you end up with the cursor where it was when you started), and the output hooks, in lines 2370-2490.

Lines 2510 and 2520 test the enable flag and exit if it's zero. Lines 2530-2650 simulate a PR#SLOT, and kill video echo. Change SLOT in Line 1060 if printer isn't in slot 1, and change NOAIO to 0 in Line 1080 if you have an AIO interface, which is code from Steve Knouse.

Lines 2670-2720 start the dump with screenline specified by WINTOP. In Lines 2730-2820 we handle 80-column dumps. This is good only for //e or //c or IIgs using firmware 80-column routines. There are too many video interface cards that work in different ways to support them all. Some cards have their own dump utilities, so you may not be out of luck if you have a non-standard interface. In line 2810 I use $FF to flag the X-register if 80-columns are active. Otherwise, lines 2840-2870 in MORCHR will end with X in range 0-23. That way in Lines 3090 and 3100 if X gets incremented to 0 then branch back to MORCHR to get even column character, so 2*40=80 characters/line get dumped. This trick of using a register value out of the range you would normally get in the rest of your code is a handy way of saving bytes.

In Line 2840 PRFN gets tested. 0->LORES, and 1->TEXT. If LORES, still only do a mixed mode dump of 20 LORES lines and 4 TEXT lines. Replace Line 2880 by 2 NOP's ($EA) to do full-screen LORES. Note that if you're in 80-column mode, the odd columns are dumped as text regardless of PRFN, since 2820 skips MORCHR. So this dump doesn't do double-LORES. Also, your LORES-creating program should start with TEXT:HOME:GR since TEXT:HOME clears even 80-columns to spaces, whereas GR only clears 40 columns to nulls. If you just do GR, some text may be left in the aux memory text pages, and you'll see it when you do a LORES dump.

So a LORES dump in 80-column mode still dumps only 40 column LORES, spaced out (if you did TEXT:HOME:GR) to match the 4 lines of 80-column text at the bottom.

In Lines 2890-3010 a quasi-40-by-40 resolution for LORES is done by using a space for both LORES blocks off, a caret for top LORES block on, an underline for bottom LORES block on, and an X for both LORES blocks on. This should work for most printers, but feel free to use other characters or symbols, especially if your printer has graphics. The resulting dump is good for monochrome LORES pictures, and especially Bar Charts. Multi-colored LORES and even abstract monochrome patterns lose a lot in the 'translation' to print.

Note especially how 2910 turns off the lo-nibble, while the exclusive-or in 2930 turns off the hi-nibble and turns back on the lo-nibble. Exclusive-or is a handy opcode to learn, and to save bytes by using. Also, the BIT opcode is used in Lines 2960, 2980, and 3000 as a 2-byte NOP to skip the following 2 bytes. BIT changes the N,V, and Z flags, but since 3020 does another CMP anyway, this dosn't matter here. Another 'gotcha' is to avoid inadvertently toggling a soft-switch in the $C000-C0FF range via BIT. I did that once and it took me a long time to figure out!

Lines 3020-3050 make sure a value of $A0 or greater is sent to printer, to avoid control or inverse or flashing characters. Line 3060 then masks off the MSB to avoid graphics in EPSON printers. Lines 3080-3190 send the character to the printer and do loop checking for 80-columns, line length, and WINBOT screen checking.

Lines 3210-3410 restore everything saved upon entry to DUMP. If FLAG was set to show we came thru TRAP, it's cleared in case next time we don't. Besides, if we came thru TRAP, we're in input mode and need to JMP RDKEY to get the next keystroke. If we didn't come thru TRAP, we can just exit via RTS.

Some Demonstrations

So much for the dump. Now it's time for some demo programs. The following, which I call CHARDEMO, shows the active character set, including MouseText if you're in 80-column mode. //e's, c's, and IIgs's have different character sets for 40 and 80 column modes, and you'll see that most printers match the 40-column character set best for INVERSE and FLASHing characters. So what you dump isn't always what you see! If you avoid MouseText and INVERSE and FLASH, you'll do all right except that most printers use code $7F=127 as 'delete previous char' instead of a checkerboard. So if you don't see a tilde (code $7E=126) or a checkerboard, you'll know what happened.

```
5  REM CHARDEMO
6  PRINT  CHR$ (4)"BLOAD SCRNDUMP.PLUS"
7  CALL 692: REM INSTALL HOOKS
10  INVERSE
20  PRINT  CHR$ (27);"@ABCDEFGHIJKLMNOPQRS
    TUVWXYZ[\]^_"; CHR$ (24)
30  NORMAL
40  GOSUB 1000
50  INVERSE : GOSUB 1000: NORMAL
60  FLASH : GOSUB 1000: NORMAL
70  END
1000  FOR I = 32 TO 127: PRINT  CHR$ (I);: NEXT
1010  PRINT : RETURN
```

TESTGR does a LORES dump by CALLing DUMP directly. Try in 40 and 80-column mode to see the difference.

```
5  TEXT : HOME : GR
10  COLOR= 15
20  FOR I = 0 TO 39
30  VLIN I,39 AT I: NEXT
40  PRINT "THIS IS A TEST."
50  POKE 9,0: CALL 802: POKE 9,1
60  REM ASSUNES M/L BLOADED & PARAMETERS SET
```

TEST40 and TEST80 merely show that no top lines or characters get dropped.

```
 5  TEXT : HOME
10   FOR I = 1 TO 22
20   PRINT "LINE NUMBER ";I;: HTAB 16
25   PRINT "THIS IS A TEST."
30   NEXT
```

```
 5  TEXT : HOME
10   FOR I = 1 TO 22
20   PRINT "LINE NUMBER ";I;: HTAB 16
25   PRINT "THIS IS A TEST.";
26   FOR J = 1 TO 40: PRINT "A";: NEXT : PRINT
30   NEXT
```

LORES.PIC, adapted from a David Thornburg program to do LORES and not HIRES patterns, allows you to experiment with patterns. Try Inputs x=0,y=0,s=field size=10, and 2 colors to see an abstract monochrome pattern-it loses a lot in the translation to print.

```
 5  REM LORES.PIC/DAVID THORNBURG/A+/DEC.86&JUN.87
 6  REM ASSUMES M/L BLOADED & PARAMETERS SET
 7  POKE 9,0: REM FOR LORES DUMP
10   PRINT "ENTER STARTING X-VALUE ";
20   INPUT A
30   PRINT "ENTER STARTING Y-VALUE ";
40   INPUT B
50   PRINT "ENTER FIELD SIZE ";
60   INPUT S
70   PRINT "ENTER NUMBER OF COLORS (2-16) ";
80   INPUT N
90   TEXT : HOME : GR
100   FOR I = 0 TO 39
110   FOR J = 0 TO 39
120 X = A + (S * (I - 20) / 100)
130 Y = B + (S * (J - 20) / 100)
140 Z = 10 *  SIN (X * X) + 10 *  SIN (Y * Y)
160   COLOR= (Z - (N *  INT (Z / N)))
170   PLOT I,J
180   NEXT J
190   NEXT I
200   CALL 802: POKE 9,1: REM DUMP, THEN BACK TO TEXT
DEFAULT
```

Try changing Line 140 to Z= X*X + Y*Y and use inputs 0,0,25,2 to see circles. If the pattern isn't too abstract, the resulting dump is ok. Also try 140 Z = X*Y and inputs 0,0,20,2 to see hyperbolas. It's a neat program to play with, but unfortunately abstract and/or multi-colored patterns go past the capabilities of the simplistic LORES dump here.

```
                1000 *SAVE S.SCRNDUMP.PLUS
                1010 *-------------------------------
                1020 * See AAL/SEP.83/P.22-24/Steve Knouse
                1030 * and NIBBLE/#3/P.19&40/R.M. Mottola
                1040 * ...modified 5/10/87 Louis Pitz
                1050 *-------------------------------
01-             1060 SLOT    .EQ 1          PRINTER SLOT#
01-             1070 DOS     .EQ 1          DOS3.3 ACTIVE
01-             1080 NOAIO   .EQ 1          GENERIC INTERFACE

                1100 *-------------------------------
06-             1110 PRFLAG .EQ $6          0->DISABLE, 1->ENABLE
07-             1120 WINTOP .EQ $7          TOP OF WINDOW TO PRINT
08-             1130 WINBOT .EQ $8          BOTTOM
09-             1140 PRFN   .EQ $9          0->LORES, 1->TEXT
EB-             1150 PTR    .EQ $EB         USE FOR RELOCATING
24-             1160 CH     .EQ $24
28-             1170 BASL   .EQ $28
36-             1180 CSWL   .EQ $36
37-             1190 CSWH   .EQ $37
38-             1200 KSWL   .EQ $38
39-             1210 KSWH   .EQ $39
ED-             1220 FLAG   .EQ $ED
EE-             1230 LINCTR .EQ $EE
                1240 *-------------------------------
0100-           1250 STACK  .EQ $100        USE FOR RELOCATING
                1260 *-------------------------------
03D0-           1270 DOS.WARM   .EQ $3D0
03EA-           1280 DOS.HOOK   .EQ $3EA
AA55-           1290 DOSKSW .EQ $AA55       DOS ACTIVE->TRUE KSW
BE32-           1300 VECTIN .EQ $BE32       PRODOS ACTIVE->TRUE KSW
9A17-           1310 RSTINT .EQ $9A17
E003-           1320 BASIC2 .EQ $E003
                1330 *-------------------------------
```

```
0578-           1340 NOVID   .EQ $578
C01F-           1350 COL80   .EQ $C01F     80-COLUMN ON IF MSB=1
C054-           1360 REGRAM  .EQ $C054     SELECT MAIN RAM TEXT PAGE
C055-           1370 AUXRAM  .EQ $C055     ...AUX RAM
FBB3-           1380 IDBYTE  .EQ $FBB3     6->//e OR c OR gs
FBC1-           1390 BASCAL  .EQ $FBC1
FC22-           1400 VTAB    .EQ $FC22
FD0C-           1410 RDKEY   .EQ $FD0C
FD1B-           1420 KEYIN   .EQ $FD1B
FD8E-           1430 CROUT   .EQ $FD8E
FDED-           1440 COUT    .EQ $FDED
FE95-           1450 OUTPRT  .EQ $FE95
FF58-           1460 RTRN    .EQ $FF58
                1470 *--------------------------------
                1480 * First find out 'WHERE AM I?'
                1490 * See "Assembly Lines:  The Book", chapter 14
                1500 *   by Roger Wagner
                1510 *--------------------------------
                1520         .OR $2B4      FIT $2B4.3CF A$2B4,L$11C [A692,L284]
                1530         .TF SCRNDUMP.PLUS
02B4- 08        1540 DISINT  PHP           SAVE INTERRUPT STATUS
02B5- 78        1550         SEI           DISABLE INTERRUPTS
02B6- 20 58 FF  1560 START   JSR RTRN      PUT START+2 ON STACK
02B9- BA        1570         TSX           NOW GET STACK POINTER
02BA- BD 00 01  1580         LDA STACK,X   GET (START+2)-HI BYTE
02BD- 85 EC     1590         STA PTR+1
02BF- CA        1600         DEX
02C0- BD 00 01  1610         LDA STACK,X   GET (START+2)-LO BYTE
02C3- 85 EB     1620         STA PTR
02C5- 28        1630         PLP           RESTORE INTERRUPT STATUS
02C6- A0 42     1640         LDY #SETRAP-START-1 OFFSET (SETRAP+1)-(START+2)
02C8- 18        1650         CLC
02C9- 69 4C     1660         ADC #TRAP-START-2   OFFSET (TRAP)-(START+2)
02CB- 91 EB     1670         STA (PTR),Y   PUT IN SETRAP+1
02CD- A5 EC     1680         LDA PTR+1
02CF- 69 00     1690         ADC #0        IN CASE CROSS PAGE BOUNDARY
02D1- C8        1700         INY
02D2- C8        1710         INY           SO Y=OFFSET (SETRAP+3)-(START+2)
02D3- 91 EB     1720         STA (PTR),Y   PUT IN SETRAP+3
                1730      .DO DOS       IF DOS ACTIVE
02D5- AD 55 AA  1740         LDA DOSKSW    SAVE INPUT HOOKS INSIDE TRAP
                1750      .ELSE         IF PRODOS
                1760         LDA VECTIN
                1770      .FIN
02D8- A0 54     1780         LDY #TRAP+6-START OFFSET (TRAP+8)-(START+2)
02DA- 91 EB     1790         STA (PTR),Y
02DC- A0 4D     1800         LDY #TRAP-START-1 OFFSET (TRAP+1)-(START+2)
02DE- 91 EB     1810         STA (PTR),Y
02E0- C8        1820         INY           NOW OFFSET (TRAP+2)-(START+2)
                1830      .DO DOS       IF DOS ACTIVE
02E1- AD 56 AA  1840         LDA DOSKSW+1
                1850      .ELSE         IF PRODOS
                1860         LDA VECTIN+1
                1870      .FIN
02E4- 91 EB     1880         STA (PTR),Y
02E6- A0 56     1890         LDY #TRAP+8-START OFFSET (TRAP+10)-(START+2)
02E8- 91 EB     1900         STA (PTR),Y
02EA- A2 00     1910         LDX #0        SET DEFAULT VALUES
02EC- 86 07     1920         STX WINTOP    TOP OF SCREEN=LINE 0
02EE- 86 ED     1930         STX FLAG      CLEAR FLAG AT START
02F0- E8        1940         INX
02F1- 86 06     1950         STX PRFLAG    ENABLE ROUTINE
02F3- 86 09     1960         STX PRFN      SET FOR TEXT
02F5- A2 17     1970         LDX #23       SET BOTTOM SCREEN
02F7- 86 08     1980         STX WINBOT    TO LINE 23
                1990 *--------------------------------
                2000 * NOW FOR SETRAP -SET INPUT HOOKS
                2010 * TO POINT AT 'TRAP' OR INPUT FILTER
                2020 * NOTE HOW FIRST SECTION MODIFIES (SETRAP+1)
                2030 * AND (SETRAP+3) TO POINT AT TRAP!
                2040 *--------------------------------
02F9- A2 04     2050 SETRAP LDX #TRAP
02FB- A0 03     2060         LDY /TRAP
                2070      .DO DOS       IF DOS ACTIVE
02FD- 86 38     2080         STX KSWL
02FF- 84 39     2090         STY KSWH
0301- 4C EA 03  2100         JMP DOS.HOOK
                2110      .ELSE         PRODOS
                2120         STX VECTIN
                2130         STY VECTIN+1
                2140         RTS
                2150      .FIN
```

```
                        2160 *--------------------------------
0304- 20 1B FD           2170 TRAP    JSR KEYIN     GET KEYPRESS
0307- C9 94              2180         CMP #$94      CTRL-T FOR TERMINATE?
0309- D0 0E              2190         BNE .1        NO
030B- A2 1B              2200         LDX #KEYIN    RESET INPUT HOOKS
030D- A0 FD              2210         LDY /KEYIN    WHICH HAVE BEEN SAVED
030F- 86 38              2220         STX KSWL
0311- 84 39              2230         STY KSWH
                         2240      .DO DOS       IF DOS ACTIVE
0313- 20 EA 03           2250         JSR DOS.HOOK PASS TO DOS
0316- 4C D0 03           2260         JMP DOS.WARM AND EXIT
                         2270      .ELSE         PRODOS
                         2280         JSR RSTINT    PASS TO PRODOS
                         2290         JMP BASIC2    AND EXIT
                         2300      .FIN
0319- C9 90              2310 .1      CMP #$90      CTRL-P FOR PRINT?
031B- F0 01              2320         BEQ .2        YES->BRANCH
031D- 60                 2330         RTS           NO-> EXIT
031E- A9 FF              2340 .2      LDA #$FF      RESET FLAG TO SHOW
0320- 85 ED              2350         STA FLAG      HAVE COME THRU TRAP
                         2360 *--------------------------------
0322- 08                 2370 DUMP    PHP           SAVE INTERRUPT STATUS
0323- 78                 2380         SEI           DISABLE INTERRUPTS
0324- 48                 2390         PHA           SAVE A,X,Y
0325- 8A                 2400         TXA
0326- 48                 2410         PHA
0327- 98                 2420         TYA
0328- 48                 2430         PHA
0329- A5 24              2440         LDA CH        SAVE CH
032B- 48                 2450         PHA
032C- A5 36              2460         LDA CSWL      SAVE OUTPUT HOOKS
032E- 48                 2470         PHA
032F- A5 37              2480         LDA CSWH
0331- 48                 2490         PHA
                         2500 *
0332- A5 06              2510         LDA PRFLAG    ROUTINE ENABLED?
0334- F0 7A              2520         BEQ RESTOR    IF NOT, GET OUT
0336- A9 01              2530         LDA #SLOT     COLD START BOARD
0338- 20 95 FE           2540         JSR OUTPRT    IN SLOT
                         2550      .DO NOAIO     GENERIC INTERFACE
033B- A9 89              2560         LDA #$89      KILL VIDEO ECHO
033D- 20 ED FD           2570         JSR COUT      VIA
0340- A9 CE              2580         LDA #"N       CTRL-I"N"
0342- 20 ED FD           2590         JSR COUT
                         2600      .ELSE         SSM AIO INTERFACE
                         2610         LDA #$80
                         2620         JSR COUT
                         2630         LDX SLOT
                         2640         STA NOVID,X
                         2650      .FIN
                         2660 *
0345- 20 8E FD           2670         JSR CROUT     START ON A NEW LINE
0348- A5 07              2680         LDA WINTOP
034A- 85 EE              2690         STA LINCTR    COUNT SCREEN LINES
034C- 20 C1 FB           2700 NXTLN   JSR BASCAL    GET ADDR OF LINE
034F- A0 00              2710         LDY #0        START W/ 1ST CHAR (0-TH)
0351- 84 24              2720         STY CH        SET CH=0 TO START LEFT EDGE
0353- 2C 1F C0           2730 NXTCHR BIT COL80      80-COL ON?
0356- 10 13              2740         BPL MORCHR    NO->BRANCH
0358- AD B3 FB           2750         LDA IDBYTE    //e OR c OR gs?
035B- C9 06              2760         CMP #6
035D- D0 0C              2770         BNE MORCHR    NO->BRANCH
035F- 8D 55 C0           2780         STA AUXRAM    READ ODD COLUMN
0362- B1 28              2790         LDA (BASL),Y
0364- 8D 54 C0           2800         STA REGRAM    READY FOR EVEN COLUMN
0367- A2 FF              2810         LDX #$FF      SHOW 80-COL ON
0369- D0 23              2820         BNE INVCHK    SEND ODD COLUMN CHAR
036B- B1 28              2830 MORCHR LDA (BASL),Y
036D- A6 09              2840         LDX PRFN      LORES OR TEXT?
036F- D0 1D              2850         BNE INVCHK    IF TEXT THEN BRANCH
0371- A6 EE              2860         LDX LINCTR    SO NOW LORES
0373- E0 14              2870         CPX #20       BUT DO AT MOST 20 LORES & 4 TEXT
0375- 10 17              2880         BPL INVCHK    SO TEXT IF PAST LINE 20
0377- C9 00              2890         CMP #0        ZERO GRAPHICS?
0379- F0 0B              2900         BEQ .1        YES, USE SPACE
037B- 29 F0              2910         AND #$F0      HI-NIBBLE=LO-LORES BLOCK
037D- F0 0A              2920         BEQ .2        0->USE ^, HI-LORES BLOCK ON
037F- 51 28              2930         EOR (BASL),Y LO-NIBBLE=HI-LORES BLOCK
0381- F0 09              2940         BEQ .3        0->USE _, LO-LORES BLOCK ON
```

```
0383- A9 D8     2950          LDA #"X       USE X, BOTH BLOCKS ON
0385- 2C        2960          .HS 2C        (BIT TO SKIP NEXT 2 BYTES)
0386- A9 A0     2970 .1       LDA #$A0      SPACE FOR ZERO
0388- 2C        2980          .HS 2C
0389- A9 DE     2990 .2       LDA #"^       CARET FOR HI-LORES BLOCK ONLY
038B- 2C        3000          .HS 2C
038C- A9 DF     3010 .3       LDA #"_       UNDERLINE FOR LO-LORES BLOCK ONLY
038E- C9 A0     3020 INVCHK   CMP #$A0      INVERSE OR FLASHING?
0390- B0 04     3030          BCS REGCHR    NO, SO REGULAR CHAR
0392- 69 40     3040          ADC #$40      YES, ALTER BITS 6 & 7
0394- D0 F8     3050          BNE INVCHK    AND KEEP CHECKING
0396- 29 7F     3060 REGCHR   AND #$7F      MASK OFF HI BIT TO AVOID
                3070 *                      EPSON BLOCK GRAPHICS
0398- 20 ED FD  3080          JSR COUT      PRINT IT
039B- E8        3090          INX           80-COL ON-> FROM $FF TO 0
039C- F0 CD     3100          BEQ MORCHR    IF SO, GET EVEN COLUMN CHAR
039E- C8        3110          INY
039F- C0 28     3120          CPY #40       WHOLE LINE DONE?
03A1- 90 B0     3130          BCC NXTCHR    NO-GET NEXT CHAR
03A3- 20 8E FD  3140          JSR CROUT     END OF LINE
03A6- E6 EE     3150          INC LINCTR    GOTO NEXT LINE #
03A8- A5 EE     3160          LDA LINCTR
03AA- C5 08     3170          CMP WINBOT    AT BOTTOM SCREEN YET?
03AC- 30 9E     3180          BMI NXTLN     NO-GOTO NEXT LINE
03AE- F0 9C     3190          BEQ NXTLN     YES-GET LAST LINE
                3200
03B0- 68        3210 RESTOR   PLA           RESTORE OUTPUT HOOKS
03B1- 85 37     3220          STA CSWH
03B3- 68        3230          PLA
03B4- 85 36     3240          STA CSWL
03B6- 68        3250          PLA           RESTORE CH
03B7- 85 24     3260          STA CH
03B9- 20 22 FC  3270          JSR VTAB      AND LINE
03BC- 68        3280          PLA           RESTORE Y,X,A
03BD- A8        3290          TAY
03BE- 68        3300          PLA
03BF- AA        3310          TAX
03C0- A5 ED     3320          LDA FLAG      DID WE COME THRU TRAP?
03C2- D0 03     3330          BNE CLEAR     YES->BRANCH
03C4- 68        3340          PLA           NO-> RESTORE (A)
03C5- 28        3350          PLP           RESTORE INTERRUPT STATUS
03C6- 60        3360          RTS           AND EXIT.
03C7- A9 00     3370 CLEAR    LDA #0        CLEAR FLAG
03C9- 85 ED     3380          STA FLAG
03CB- 68        3390          PLA           RESTORE (A)
03CC- 28        3400          PLP           RESTORE INTERRUPT STATUS
03CD- 4C 0C FD  3410          JMP RDKEY     GET NEXT KEY-JMPS TO (KSWL)
                3420 *--------------------------------
```

It's 1988, and ProDOS Thinks it's 1982....Bob Sander-Cederlof

If you are still using ProDOS 1.1.1, and you have some sort of clock card such as Thunderclock, TimeMaster, or any other "standard" ProDOS clock, you have a problem. Apple built this bug into ProDOS, and they came out with the new versions (they call it ProDOS-8 version 1.4 now) just in time.

In my article about the clock driver in the November 1983 issue of AAL (pages 25-28), I discussed the problem. It seemed a little more remote at the time. Apple based ProDOS on the Thunderclock, even though that device does not keep track of the year. The ProDOS clock driver reads the Month, Day, and Day of Week information and does some arithmetic to determine which of six years could produce that day of week on the corresponding month and day. ProDOS 1.1.1 and earlier versions could produce dates from 1982 through 1987. When 1988 rolled around a few weeks ago, hundreds of thousands of Applers around the world slipped back in time to 1982.

And it is not funny! Some programs will not let you operate if the dates are not correct!

Well, there are at least four ways around the problem. You can remove your clock card, and type the date in manually wherever it is really needed. Not very nice.

Or, you can get the up-to-date version of ProDOS, now called ProDOS-8 Version 1.4. You can get it, and then you can copy it to every floppy (both 3 1/2 and 5 1/4), to every RamFactor, to every hard disk in sight. This is tedious, but it is the best solution. If you have a friendly dealer, you can get it from the IIgs system disk. But don't copy the file named PRODOS from this disk (that is only a loader now). Instead, copy the file named P8 from the subdirectory SYSTEM. P8 is a longer file than version 1.1.1 of PRODOS was, so if you use BSAVE to put it on your disks be sure to specify the L parameter. Something like this should do the trick:

> Boot any ProDOS disk, preferably one with version 1.4 so the correct dates will get into the file directories you are updating. Get into the S-C Macro Assembler or Applesoft. With the latest IIgs system disk in your drive, type:
>
> ```
> BLOAD SYSTEM/P8,TSYS,A$2000
> ```
>
> Now put the disk you want to update into a drive, and type the following. You may want to include slot and drive parameters, or set the prefix to the appropriate value for a ram disk or hard disk.
>
> ```
> UNLOCK PRODOS
> BSAVE PRODOS,TSYS,A$2000,L$3C7D
> LOCK PRODOS
> ```

A third approach saves you a trip to the dealer. You can simply PATCH the copies of ProDOS version 1.1.1 to give you the correct year. When you BLOAD the file named PRODOS at $2000, the six-year table is at $4F76. If you look there now you will find the following bytes:

```
4F76: 54 54 53 52 57 56 55
```

These correspond to the years 1984, 1984, 1983, 1982, 1987, 1986, and 1985. Notice that 1984, being a leap year, takes up two of the values. Patch these seven bytes, using the monitor, as follows:

```
4F76:5A 59 58 58 57 56 5B
```

The table now includes the years from 1986 through 1991. If you want 1992 in there also, substitute 5C where I have 57 and 56 above. Both 1988 and 1992 are leap years, so they both take two table positions. When ProDOS 1.4 was released it was still 1987, so there was not room for 1992 in the table.

A fourth possible solution was suggested by reader Garth O'Donnell. You can replace the clock driver inside ProDOS with one that reads the year directly from your clock card! This is what happens when you boot Version 1.4 in a IIgs, because P8 senses that you are in a IIgs and plugs in a different driver. But if you are still using an older Apple, as most of us are, you can modify the PRODOS file to load an intelligent driver for your own clock card. Of course, if you are using a Thunderclock, the driver with the above patches is the best you can do. But if you have a TimeMaster, as Garth does, you can use a program like he wrote.

I decided to try my hand at modifying the standard clock driver so that it uses the year information in the TimeMaster. The following program is derived directly from the standard driver, with as few modifications as possible. It still resides in the ProDOS SYS file at $4F00, but it is a lot shorter. (Maybe you can think of something useful to do with the extra 45 bytes!) It still depends upon the standard ProDOS loader to plug in the actual slot number in lines 1260 and 1310. The major change I made was to call on the ":" instead of the "#" mode. The "#" mode is a ThunderClock mode, which does not return the year. The ":" mode is a TimeMaster mode, which does return the year.

If you have an Applied Engineering Serial Pro card, which includes a TimeMaster compatible clock, you can use the driver I wrote by making the single change as shown in the comments on line 1090. Or, maybe you could use those extra 45 bytes for a subroutine that would check which clock is in the slot and make the appropriate changes at run time.

```
4F00-                1180          .TF B.CLOCK.DRIVER
                     1000 *SAVE S.CLOCK.1988
                     1010 *--------------------------------
                     1020 *  IF THE PRODOS BOOT RECOGNIZES A TIMEMASTER,
                     1030 *  A "JMP $D742" IS INSTALLED AT $BF06 AND
                     1040 *  THE SLOT ADDRESS IS PATCHED INTO THE FOLLOWING
                     1050 *  CODE AT SLOT.A AND SLOT.B BELOW.
                     1060 *--------------------------------
                     1070 *    DEFINE CLOCK ENTRY POINT
                     1080 *--------------------------------
C108-                1090 CLOCK   .EQ $C108     <<<USE $C11D FOR AE SERIAL PRO>>>
                     1100 *--------------------------------
BF90-                1110 DATE    .EQ $BF90     $BF91 = YYYYYYYM
                     1120 *                     $BF90 = MMMDDDDD
BF92-                1130 TIME    .EQ DATE+2    $BF93 = 000HHHHH
                     1140 *                     $BF92 = 00MMMMMM
0538-                1150 MODE    .EQ $5F8-$C0 TIMEMASTER MODE IN SCREEN HOLE
                     1160 *--------------------------------
                     1170          .OR $4F00
4F00-                1180          .TF B.CLOCK.DRIVER
                     1190          .PH $D742
                     1200 *--------------------------------
                     1210 PRODOS.TIMEMASTER.DRIVER
D742- AE 50 D7       1220          LDX SLOT.B    $CN
D745- BD 38 05       1230          LDA MODE,X    SAVE CURRENT TIMEMASTER MODE
D748- 48             1240          PHA
D749- A9 BA          1250          LDA #":"      SEND ":" TO TIMEMASTER
D74B- 20 0B C1       1260          JSR CLOCK+3         SELECT TIMEMASTER MODE
D74D-                1270 SLOT.A .EQ *-1
                     1280 *--------------------------------
                     1290 *       READ TIME & DATE INTO $200...$211 IN FORMAT:
                     1300 *--------------------------------
D74E- 20 08 C1       1310          JSR CLOCK
D750-                1320 SLOT.B .EQ *-1
```

```
                    1330 *--------------------------------
                    1340 *       CONVERT ASCII VALUES TO BINARY
                    1350 *       $3E -- MINUTE
                    1360 *       $3D -- HOUR
                    1370 *       $3C -- YEAR
                    1380 *       $3B -- DAY OF MONTH
                    1390 *       $3A -- MONTH
                    1400 *--------------------------------
D751- 18            1410         CLC
D752- A2 04         1420         LDX #4
D754- A0 0C         1430         LDY #12         POINT AT MINUTE
D756- B9 03 02      1440 .1      LDA $203,Y      TEN'S DIGIT
D759- 29 0F         1450         AND #$0F        IGNORE TOP BIT
D75B- 85 3A         1460         STA $3A         MULTIPLY DIGIT BY TEN
D75D- 0A            1470         ASL             *2
D75E- 0A            1480         ASL             *4
D75F- 65 3A         1490         ADC $3A         *5
D761- 0A            1500         ASL             *10
D762- 79 04 02      1510         ADC $204,Y      ADD UNIT'S DIGIT
D765- 38            1520         SEC
D766- E9 B0         1530         SBC #$B0        SUBTRACT ASCII ZERO
D768- 95 3A         1540         STA $3A,X       STORE VALUE
D76A- 88            1550         DEY             BACK UP TO PREVIOUS FIELD
D76B- 88            1560         DEY
D76C- 88            1570         DEY
D76D- CA            1580         DEX             BACK UP TO PREVIOUS VALUE
D76E- 10 E6         1590         BPL .1          ...UNTIL ALL 5 FIELDS CONVERTED
                    1600 *--------------------------------
                    1610 *       PACK MONTH AND DAY OF MONTH,
                    1620 *--------------------------------
D770- A8            1630         TAY             MONTH (1...12)
D771- 4A            1640         LSR             00000ABC--D
D772- 6A            1650         ROR             D00000AB--C
D773- 6A            1660         ROR             CD00000A--B
D774- 6A            1670         ROR             BCD00000--A
D775- 05 3B         1680         ORA $3B         MERGE DAY OF MONTH
D777- 8D 90 BF      1690         STA DATE        SAVE PACKED DAY AND MONTH
                    1700 *--------------------------------
D77A- A5 3C         1710         LDA $3C         YEAR
D77C- 2A            1720         ROL             MERGE TOP MONTH BIT
D77D- 8D 91 BF      1730         STA DATE+1      YYYYYYYM
                    1740 *--------------------------------
D780- A5 3D         1750         LDA $3D         GET HOUR
D782- 8D 93 BF      1760         STA TIME+1
D785- A5 3E         1770         LDA $3E         GET MINUTE
D787- 8D 92 BF      1780         STA TIME
D78A- 68            1790         PLA             RESTORE TIMEMASTER MODE
D78B- AE 50 D7      1800         LDX SLOT.B      GET $CN FOR INDEX
D78E- 9D 38 05      1810         STA MODE,X
D791- 60            1820         RTS
```

S-C Software Corporation

2331 Gus Thomasson, Suite 125, P.O. Box 280300, Dallas, Texas 75228 (214) 324-2050

Our software is not only unlocked and fully copyable...we often provide the complete source code on disk, allowing you to understand how it works and make your own personal extensions.

S-C Cross Reference Utility: A support program which works with the S-C Macro Assembler to generate an alphabetized listing of all labels in an assembly language source program, showing with each label the line number where it is defined and all the line numbers containing references to that label. Works with multi-file source programs. The ProDOS version automatically follows the .IN or .INB ("include file") directives and assigns a code-letter to each included file in the cross reference listing. We include the complete source code for this amazingly fast program, in both DOS and ProDOS versions, for only $50.

S-C DisAssembler: An intelligent two-pass disassembler which works in conjunction with the ProDOS version of the S-C Macro Assembler. Converts files of 6502 or 65C02 machine language programs into meaningful source code files complete with labels. Driven by a script you write in a convenient "disassembly" language, allowing you to define label names of your choice, process all or part of input files, and many other options. A built-in option produces comment lines before each label showing all reference points to that label. Comes with complete commented source code and easy-to-understand documentation, for only $50.

Laumer Research Full Screen Editor: Integrates with the built-in line-oriented editor in the S-C Macro Assembler to provide a powerful full-screen editor for your assembly language source files. Drivers for Videx, STB80, and Apple //e (and //c) 80-column boards are included, as well as a 40-column version. Comes with complete source code, with both DOS and ProDOS versions, for only $49.

ProVIEW, by Doug McComsey: A professional tool for "zapping" ProDOS disks and memory. On an Apple //c, //e, or //gs in 80-column mode ProVIEW gives you a 256-byte window into RAM, ROM, ProDOS files, and disk blocks. You can examine, modify, and update any portion of RAM, a file, or a disk block. Operates from a series of menu screens much like Appleworks, with complete HELP available at every step. Only $20!

S-C Word Processor: The one we use for manuals, letters, our monthly newsletter, and whatever. 40-columns only (II and II+ require lower-case display mod and shift-key mod). Works with standard DOS text files, but at super fast speeds (100 sectors in 7 seconds). No competition for Word Perfect, but you get complete source code! $50, DOS only. We also will include at no extra charge two additional disks. One contains the source and object code for an 80-column //e, //c, //gs. We use this one, but it has a few cosmetic un-features. Sorry, still only in DOS 3.3. The other disk contains the source and object code for a 40-column version which runs under ProDOS, on any ProDOS machine. We use both of these, but they are not as polished as the original. All for only $50!

S-C Double Precision Supplement for Applesoft: Two complete packages for the price of one! Only $50 gets you the complete commented source code for both DPFP and DP18, with documentation. DPFP gives you 21-digit floating point precision for INPUT, PRINT, and +-*/ operations. DP18 is more complete, offering all of the math operations and functions, formatted I/O conversions, and 18-digit decimal floating point. DP18 is ideal for writing programs which deal with money. Both packages use the "&" command in Applesoft to add the new capabilities, without losing any of the old. DP18 includes both DOS and ProDOS versions, DPFP is DOS only. Detailed internal documentation for DP18 was published in 12 consecutive issues of "Apple Assembly Line", and is available for an additional $18.

Apple Assembly Line (ISSN 0889-4302) is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $18 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $14 postage for other countries. Back issues are $1.80 each (other countries inquire for postage). A subscription to the newsletter and the Monthly Disk containing all source code is $64 per year in the US, Canada and Mexico, and $87 to other countries.